A Constraint-based Flight Control System Architecture for UAVs using the iTaSC Framework*

J. Verbeke[1], J. Vantilt[2], D. Vanthienen[2], M. Vochten[2], S. Debruyne[1] and J. De Schutter[2]

Abstract— **(Semi-) autonomous complex UAV missions, such as inspection or search-and-rescue in uncertain dynamic environments, require obstacle avoidance and operator shared control. Combining humans' cognitive abilities with fast automation is the key for such missions. This paper presents a flight control system architecture based on the instantaneous Task Specification using Constraints (iTaSC) methodology and software framework. iTaSC is a flexible constraint-based programming approach that generates a robot motion at runtime which automatically derives the input for a low-level controller taking into account constraints and intentions from the operator, obstacles and mission constraints. This setup is experimentally validated by navigating a multirotor UAV safely through a GPS-denied corridor using (intuitive) shared control with a pilot. In addition to the pilot's commands, automatic obstacle avoidance and object tracking are performed real-time through various onboard sensors and with limited onboard computational power.**

Keywords: Human-robot Shared Control, Constraint-based Programming, Collision Avoidance, Reactive and Sensor-Based Planning, Aerial Robotics

I. INTRODUCTION

UAVs already perform many (easy) tasks such as video shoots, aerial photography, mapping and even 3D modelling of infrastructure either manually controlled or through predefined GPS trajectories. Yet, future complex applications such as inspection and search-and-rescue [1][2], are performed in undefined dynamic environments. These (semi-)autonomous missions require obstacle avoidance and shared control for the pilot especially in confined areas, both indoor (GPS-denied areas) and outdoor. Creating a flight control system architecture that achieves the flexible integration of all these different tasks and sensor types is challenging. In this paper we will present a novel modular and generic architecture that allows combining these different tasks by solving them together, in real-time, as a *constraint-based optimization problem* using the instantaneous Task Specification using Constraints (iTaSC) methodology [3] and accompanying open-source software framework [4]. This methodology allows user-friendly and intuitive shared control, easy and fast tuning of tasks and (de-)activating these tasks automatically at runtime while retaining physically interpretable results and predictable behaviour. The computational power required for iTaSC is low enough to perform all computations on board.

Most published research focusses on fully autonomous perception and control [5], yet are currently not robust nor mature enough to evolve into off-the-shelve solutions for complex missions. Shared control by a (safety) pilot is the best intermediate solution and combines the advantages of human piloting and automation [6]. The synergy of both simplifies flying a UAV by providing the operator assistance in complex environments and tasks, while preserving the operator's flexibility and cognitive capabilities. In addition, most national regulations require, for the foreseeable future, a safety pilot able to intervene at each phase of the flight [7].

This paper's novelty lies in the implementation and experimental validation of a multirotor flight system architecture where, using the iTaSC framework, tasks such as shared control, collision avoidance and camera object tracking are performed simultaneously using only onboard sensors

*This work was supported by KU Leuven FWO-project G.0404.10 and research group ProPoLiS.

[1]Authors are with the Faculty of Engineering Technology, KU Leuven, 3001 Leuven, Belgium firstname.lastname@kuleuven.be

[2]Authors are with the Department of Mechanical Engineering PMA, KU Leuven, 3001 Leuven, Belgium firstname.lastname@kuleuven.be

and computational power. The iTaSC software framework allows a systematic integration of these different tasks and sensors while resolving possible conflicts between the tasks. The validation use-case scenario is a semi-autonomous indoor (GPS-denied) mission through a confined corridor. This requires shared control, pointing an onboard camera at a given visual target (AR marker) for camera object tracking and incorporating obstacle avoidance using only onboard (sonar and infrared) sensors and computational power. The use-case assumes the operator to navigate the UAV, based only on visual clues from a forward pointing camera. Moreover, absolute position information of the UAV in the environment is not required to accomplish the tasks.

This paper describes the flight control architecture and implementation of iTaSC on UAVs based on this use-case. First, Section II gives an overview of related work and a description of iTaSC. Then, Section III shows the overall UAV hard- and software architecture of the use-case, while Section IV presents the iTaSC implementation. Section V shows the experimental validation of the approach, while Section VI discusses its results. Finally, Section VII states the conclusions and outlines future work.

II. RELATED WORK

The main challenges for safe navigation are to provide the UAV with an apt level of awareness of its surroundings and the ability to navigate through the surroundings. These challenges are common to all mobile robots, such as ground vehicles [8], fixed-wing UAVs [9] or rotary-wing UAVs [10]. Technologies to tackle these challenges include obstacle detection and avoidance, robot localization and path planning. They rely on the perception of the environment using sensors including sonars [11], lidars [12], camera(s) [13], laser scanners [14] or other sensors.

The most popular method for indoor UAV navigation is Simultaneous Localization And Mapping (SLAM) [15] using data-rich sensors where the robot simultaneously senses the environment to build up a map of its surroundings and locates itself in that environment. We purposefully choose not to use SLAM, data-rich sensors or adaptive path planning to keep computational power and UAV weight low. Nonetheless, this is not a limitation of the iTaSC approach. In future research, tasks such as path planning and GPS trajectory tracking will be added.

Missions may be performed manually, with shared control or fully automatically. When a fully autonomous solution is not advantageous, human-robot shared control is the better choice as it increases accuracy, safety and ease of tele-operation. Shared control implies that the human pilot/user has to share the control of the UAV with a high- and/or low-level controller. Shared control can be implemented using constraint-based programming or classical motion specification methodologies.

The following sections provide a non-exhaustive overview of research on (semi-)autonomous indoor UAV navigation, shared control and constraint-based control.

A. Shared control

Typical functionalities provided by the high- and/or low-level controller are obstacle detection and collision avoidance based on sensor information while the pilot can be seen of as a path planner providing goal poses. Different approaches and implementations to obstacle avoidance and shared control are available in literature.

Mendes [11] uses a SLAM algorithm based on sonar information and a grid occupancy map to demonstrate automatic collision avoidance based on the position of the UAV relative to obstacles such as walls. The authors combine the collision avoidance with user commands, using a decision making process to decide whether the user's commands should be overwritten,

Sa et al. [16] perform inspection of pole-like structures by assigning some degrees-of-freedom (DOF) to the autonomous controller and others to the user. The UAV is stabilised and kept in place with respect to the pole using image-based visual servoing. No mixed authority within a degree of freedom is possible.

The research discussed above assigns control to the autonomous controller and takes it away from the operator. In contrast, our approach keeps both the operator control and the autonomous control by weighting their relative importance as constraints in the solution of an optimization problem.

Stegagno et al. [13] present a UAV platform designed for haptic teleoperation that can be easily operated using velocity control in real unstructured scenarios providing safety against obstacles and relying only on onboard sensors, namely IMU and RGB-D. Obstacles are stored in a probabilistically updated local obstacle map. Computations are off-board, artificial visual features had to be introduced in the environment for pose estimation and there is no mixed authority within a degree of freedom possible.

Masone et al. [17] developed a new framework for semi-autonomous path planning for mobile robots. The framework generates off-line an initial path consisting of B-splines, which can be altered at run-time by the pilot or obstacle avoidance algorithms.

Mersha et al. [18] use a port-based solution at the energy (wrench and twist) level for haptic teleoperation of a UAV. Through dynamic equations and visco-elastic coupling, they apply a desired setpoint to the UAV ensuring passivity, in other words preventing an unstable system. The calculations are performed off-board and with external sensors.

Ha et al. [19] presents a vision-based teleoperation control framework for a UAV/UGV team, which allows a remote human user to teleoperate the UAV, while the UAV tracks the UGV with an onboard camera. The shared control approach is similar to iTaSC, yet the setup only had one visual sensor, no obstacle avoidance nor experimental validation was performed.

Our research differs from previous work in one or more aspects in that we have developed a flight system that can *perform several tasks simultaneously*: shared control, obstacle avoidance and object tracking using only onboard sensors and computational power. Moreover, we validated this in an experimental setup with a UAV flying in a corridor.

B. Constraint-based programming and iTaSC

Constrained optimization pertains to the (mathematical) optimization of an objective function with respect to a number of variables subject to constraints. This objective function is a cost or energy function that needs to be minimized, or a utility function to be maximized. Constrained optimization is known in robotics for task specification and control as constraint-based programming [20]. The core idea behind the approach is to describe a robot task as a set of constraints, which do not necessarily have to be formulated in a single task frame, and a set of objective functions.

The key advantages of constraint-based programming over classical motion specification methodologies are: (i) composability of constraints: multiple constraints can be combined and they do not have to constrain the full set of degrees-of-freedom (DOF) of the robot system; (ii) reusability of constraints: constraints specify a relation between frames attached to objects that have a semantic meaning in the context of a task, therefore they can be reused with different objects or robots.

De Schutter et al. [3] present a constraint-based programming approach, denoted instantaneous Task Specification using Constraints (iTaSC). It enables a developer to generate a robot motion at runtime that complies with the constraints by automatically deriving the input for a low-level controller. Moreover, geometric uncertainties, time-independent trajectories and user-configurable task horizons can be implemented [21]. The open-source software implementation has, until now, only been used for ground vehicles or robotic arms [4]. This paper uses a Domain Specific Language (DSL) version of iTaSC which enables developers and users to easier (and hence faster) understand and (re)program constraint-based programming applications, since it provides a template and enables (manual or) automatic model verification and code generation [22].

We solve in real-time, i.e. at each time instance, a *constrained optimization problem* taking into account constraints and intentions (*tasks*) from the operator, obstacles, mission constraints, etc. This method provides fast, *reactive behaviour* to, for example, avoid obstacles or track/follow objects. It *constrains* the user's input depending on the vicinity of the objects, preventing collision and providing him/her with a more intuitive feedback, while keeping as much direct control freedom as possible available to the pilot/user. However, also *actions over a longer time* are possible. For example path

following is possible by constraining a UAV to a given path or trajectory (distance minimization). Moreover, our implementation uses a *modular and systematic approach* to (de-)activate tasks (sets of constraints) at run-time, while retaining *physically interpretable results and configuration parameters* such as control gains that are easy to tune.

III. THE USE-CASE UAV PLATFORM AND SETUP

A. Scenario

Our approach involves the presence of a human operator who has shared control and visual feedback of the UAV through a forward mounted camera that streams the video feed in real-time. This simplifies the requirements with respect to localization and path planning considerably. For localization, we only need relative localization within the corridor to prevent collisions, not a full map of the environment. Thus, small lightweight range finders are sufficient, such as sonars and an infrared range finder, to detect the structured environment for collision avoidance (walls, floor, ceiling).

We rely on the human operator to keep the nose pointed along the corridor thus only sensors measuring left, right, up- and downwards with a rather narrow field of view are required and employed. Online forward camera image processing is employed to detect single image perspective cues (marker) to complement the human operator in lateral and directional control. We implemented shared control for lateral (roll), heave (climb/descend) and directional (yaw) motion. Longitudinal (pitch) motion stays under direct user command.

B. System architecture

The flight control system architecture consists of two layers: i) low-level flight control and ii) high-level navigation and data-rich sensor processing. The flight control level's main task is to keep the UAV flying and steer it using the desired setpoints of the navigation module.

A suitable UAV and sensor setup was selected. A low-cost and sturdy DJI F450 quadcopter was selected for its compactness and high agility, paramount in flying through corridors with turbulence, that is coming from its own propeller wake (Figure 1).

To perform obstacle avoidance, several sensors are required to detect the (static and dynamic) obstacles. We use three I2C sonars (MaxSonar MB1220, range $\leqslant$ 7m) for height and left and right obstacle detection which fire at 8Hz sequentially through a custom driver. In addition, we use an infrared range finder (Sharp GP2Y0A02YK0F) for measuring the distance to the ceiling. We do not use the aft or forward mounted sonars as longitudinal control remains fully with the human operator. The flight controller is a 3DR PixHawk autopilot running customized ArduPilot ArduCopter firmware. The user commands the UAV through the PixHawk by a RC transmitter. The PixHawk reads in the sonars, infrared range finder and user inputs and feeds them to the onboard computer running iTaSC. The PixHawk communicates with iTaSC on the onboard computer through the Roscopter application converting MAVlink protocol messages (PixHawk) to ROS messages (iTaSC) and vice versa over USB-FTDI. iTaSC outputs desired velocities (lateral, heave and directional) which the PixHawk converts to motor commands through comparison with the UAV's current velocities. The latter are estimated by custom Kalman filters based on PixHawk's accelerometer, barometer and sonar data. Experiments show that PixHawk's low-level controllers can track the iTaSC velocity commands well. The forward camera is a Logitech C310 webcam mounted on board the UAV with resolution set at 640x480. The iTaSC framework software and forward camera image processing runs on an Odroid XU3 onboard computer with Ubuntu 14.04 and ROS Indigo. The remote user's laptop is connected through Wi-Fi with the onboard Odroid computer during flight to monitor the processes. The camera feed can be transmitted in real-time to the user.

IV. IMPLEMENTATION OF THE USE-CASE USING ITASC

iTaSC is a generic methodology that specifies tasks as constraints on the relative position and orientation of objects (obstacles, objects to track) with respect to the robot. We will explain the general methodology through describing the implementation of iTaSC for the UAV indoor navigation use-case. For a detailed understanding of iTaSC we

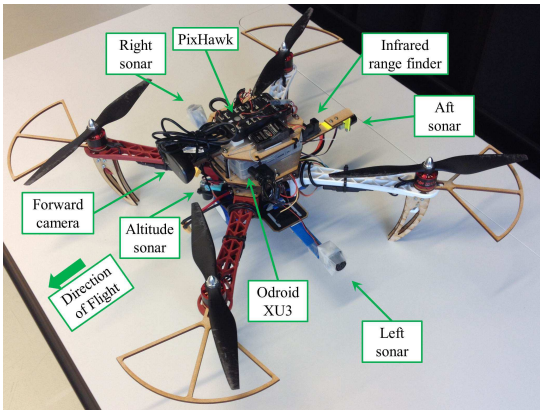


Fig. 1. The quadcopter DJI F450 used in the experiment.

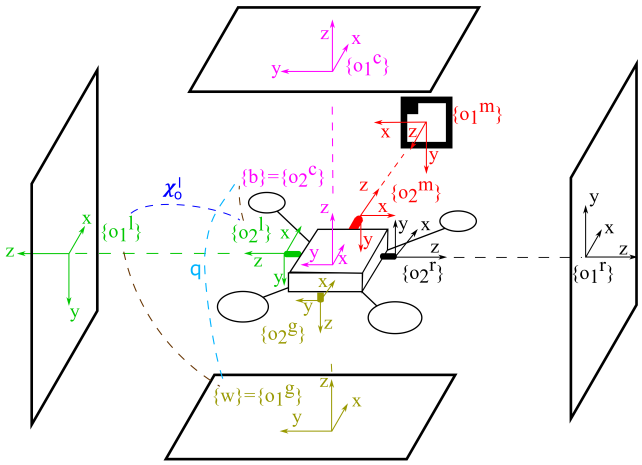


Fig. 2. Overview of the various reference frames.

refer the reader to Section II-B and the references mentioned therein.

Building and running an iTaSC application requires several steps: (i) defining a robot and objects (Section IV-A), (ii) defining the kinematic chains of the tasks (Section IV-B) and (iii) calculating the iTaSC outputs through a solver (Section IV-C). Our iTaSC application consists of: one robot, five objects i.e. four obstacle directions and one moving marker, six tasks, a world model and a solver.

A. Robots and objects

Each robot and object needs a reference frame to determine their pose w.r.t. each other and the world. Since the absolute horizontal position of the UAV w.r.t. the environment is unknown, we choose the origin of the world reference frame to instantaneously translate with the UAV base frame $\{b\}$ along x and y axis (Figure 2). As a result the UAV only translates and rotates w.r.t the world reference frame (i.e. the 'world') along the z-axis: heave and directional (yaw). The sensors of the UAV detect only the relative position of the **obstacles** (**ground, ceiling, left & right wall**) that the UAV has to avoid and the **moving marker** object that the UAV has to track (Figure 2). The obstacles detected by the sensors, have planes defined perpendicular to the relevant sensor direction and only the distance between the planes varies, except for the marker for which all DOFs vary.

B. Tasks

A task consists of the choice of a task space representation between two object frames, *constraints* on the task's kinematic loop, *controllers* enforcing the different constraints, and *set-point generators* delivering the desired reference values to the controllers [3]. Within the task space, one can choose a kinematic model, such that it simplifies the constraint definition.

The indoor corridor navigation use-case consists of six different tasks. The user needs (shared) control over the UAV: the user task. Furthermore, the UAV has to avoid collisions with its surroundings: ground, ceiling, left & right wall requiring four obstacle collision avoidance tasks, one for each obstacle. The sixth task is the marker tracking task. iTaSC has the potential to be effective as well in less-structured environments as one can add more and/or different sensors and corresponding tasks in the appropriate directions.

The **user input task** simply feeds forward the user's lateral (roll), heave (climb/descend) and directional (yaw) desired velocities, expressed in the body frame.

The **obstacle avoidance tasks** for **ground, ceiling, left and right wall** are very similar (Figure 2). Each task defines a *Cartesian space* between the object frame on the obstacle $\{o_1^o\}$, representing $\{o_1^g\}$, $\{o_1^c\}$, $\{o_1^l\}$ and $\{o_1^r\}$, and the UAV's relevant sensor frame $\{o_2^o\}$, representing $\{o_2^g\}$, $\{o_2^c\}$, $\{o_2^l\}$ and $\{o_2^r\}$. The left wall obstacle avoidance task's kinematic chain is shown in dashed lines. The sensor frames have their z-axis pointing to the sensors' measurement direction. The coordinates defining the six DOFs of the resulting kinematic

chain ${\boldsymbol{\chi_o}}^o$ can be expressed in the first object frame $\{o_1^o\}$:

$$ {\boldsymbol{\chi_o}}^o = [x^o, y^o, z^o, \phi^o, \theta^o, \psi^o]^T , \qquad (1) $$

where $[x^o, y^o, z^o]^T$ defines the $3D$ position coordinates of the obstacle with respect to the UAV's base and $[\phi^o, \theta^o, \psi^o]$ is a set of Euler-angles defining the orientation between the obstacle and the sensor frame. As both reference frames are kept parallel to each other where only z^o varies, the other values of ${\boldsymbol{\chi_o}}^o$ are zero. Since the distance is the only value of interest, only this DOF is *constrained*. An inequality constraint proportional controller is defined for the output, keeping the UAV at a safe distance D of the obstacle:

$$ {\dot{y}_d}^o = {K_p}^o(D - z^o) \text{ for } z^o \leqslant D \qquad (2) $$

The controller's desired velocity of each obstacle avoidance task ${\dot{y}_d}^o$ in relation to the distance to the obstacles y^o is shown in Figure 3(a). The safe distance D, see equation 2, is set at 1.2m for wall and ceiling tasks. The ground avoidance task differs somewhat in that it additionally constrains the UAV from climbing above the reliable sonar distance, which is around 2.5m, and therefore D is set between 0.7m and 2.5m. The task's constraint weight, in other words the importance of this task with respect to other tasks, W^o varies similar to the task's desired velocity ensuring smooth transitions and behaviour. The individual controller's desired velocities ${\dot{y}_d}^o$ and weights W^o of the each obstacle avoidance tasks in relation to the distance to their obstacles y^o are shown in Figure 3(b).

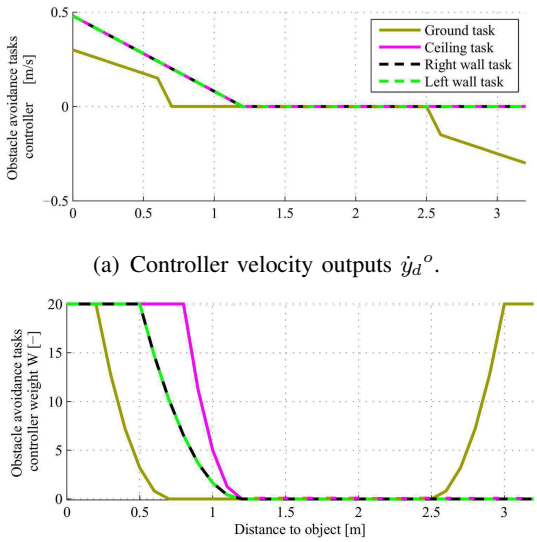
The **marker tracking task** based on the online forward camera image processing defines a *spherical space* between the object frame on the marker $\{o_1^m\}$ and the UAV's camera frame $\{o_2^m\}$. The coordinates defining the six DOFs of the resulting kinematic chain ${\boldsymbol{\chi_o}}^m$ can be expressed in the first object frame $\{o_1^m\}$:

$$ {\boldsymbol{\chi_o}}^m = [\theta^m, \phi^m, r^m, \alpha^m, \beta^m, \gamma^m]^T , \qquad (3) $$

where $[\theta^m, \phi^m, r^m]^T$ defines the spherical position coordinates of the obstacle with respect to the UAV base and $[\alpha^m, \beta^m, \gamma^m]$ is a set of Euler-angles defining the orientation between the obstacle and the UAV base. Since we only want to constrain the heading of the UAV by the video data, we only constrain θ^m:

$$ {\dot{y}_d}^m = {K_p}^m(-\theta^m) \qquad (4) $$

iTaSC automatically translates this task specification in both lateral and directional control of the UAV to align the UAV with the marker in the centre of the camera image. Depending on the weighing with the other tasks' outputs, the DOF with the least resistance will be used enabling more direct control for the pilot. The tracking task has a constant weight W^m.



(a) Controller velocity outputs ${\dot{y}_d}^o$.

(b) Controller weights W^o.

Fig. 3. The obstacle avoidance tasks controller outputs and weights.

C. Solver

The solver calculates the desired velocities $\dot{\boldsymbol{X}}_d$, to be sent to the controllable DOFs of the UAV by solving an optimization problem involving the task constraints $\dot{\boldsymbol{y}}_d$, the constraint weights $\boldsymbol{W}$, the task priorities, and the UAV joint weight matrix [22] (Figure 4). For this application, the solver calculates the desired lateral, heave and directional velocities $\dot{\boldsymbol{X}}_d$ from five constraint equations (ground, ceiling, left and right wall and marker) and the user input at 100Hz. The solver takes the weights and priorities of the different task constraints and the UAV's DOFs into account. $\dot{\boldsymbol{y}}_{\boldsymbol{d}}$ is the vector containing the desired velocities to control. $\boldsymbol{y}$ is the vector containing the coordinates to control/constrain.

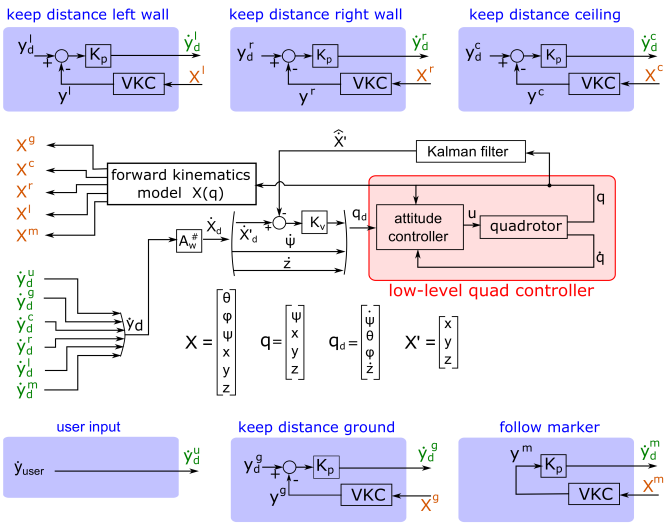


Fig. 4. The control scheme of the iTaSC indoor navigation application. The low-level controller of the UAV is represented in red. The inputs for this attitude controller are the roll, pitch and yaw angles and the altitude. The different tasks are shown in blue rectangles. They take the features $\mathbf{X}$, obtained from the kinematic chains + states $\mathbf{q}$, as inputs and deliver the desired outputs $\dot{\boldsymbol{y}}_d$. The desired task outputs are combined, weighted and send to the attitude controller.

The application's solver uses a prioritized, weighted, damped pseudo-inverse algorithm [3]. This pseudo-inverse is denoted $\boldsymbol{A}^{\#}_{W}$, where $\boldsymbol{A}$ is the augmented Jacobian relating the combined task spaces and the UAV joint space [22]. In this application, all tasks have the same priority. The obstacle avoidance weight maxima are well-chosen to achieve natural human-robot shared control. The obstacle avoidance tasks have a higher weight maxima than the user input to avoid the user flying into obstacles. As a consequence, when the user deliberately flies too close to an object his input is gradually damped by the relevant obstacle avoidance task. The marker tracking task has the same weight maximum as the user, yet the user can send larger outputs thus ensuring the user may overrule the marker tracking in case he/she desires to. The resulting desired UAV's (lateral, heave and directional) velocities $\dot{\boldsymbol{X}}_d$ are sent to the PixHawk as setpoints $\boldsymbol{q}_d$ for the attitude controller. Since the world translates with the UAV's base, pitch θ and roll ϕ of the state vector are zero. The UAV's roll and pitch inputs to the attitude controller are therefore derived from iTaSC's lateral and longitudinal velocity outputs $\dot{\boldsymbol{X}}'_d$ by comparing them with the UAV's current estimated velocities $\hat{\dot{\boldsymbol{X}}}'$ and through a proportional gain $\boldsymbol{K}_v$ transforming them into desired pitch and roll angles. THe attitude controller translates the desired roll and pitch angles together with the desired heave and directional velocities into individual motor desired rotational velocities $\mathbf{u}$.

V. EXPERIMENTAL VALIDATION

This paper provides one large experiment consisting of several phases that validate the different desired behaviours. A synchronised HD video is presented here[1]. The first phase shows the basic collision avoidance and shared control together with its fast, reactive behaviour to avoid a dynamic object. The second phase additionally includes tracking a stationary marker while following user inputs. The third phase shows tracking a dynamic marker without user inputs. Before each phase starts, the user repositions the UAV in hover at the centre of the corridor.

iTaSC is activated with the UAV in hover (@0:20) in the middle of the corridor. All tasks are active except the camera marker tracking task as no marker is present in phase 1.

A. Phase 1: basic obstacle avoidance, no marker tracking

The first phase, between 30s and 105s, shows the basic collision avoidance control, shared control and the reactive behaviour to a dynamic object (Figure 5).

Phases 1.1, @30s, **1.2,** @39s, **1.3** @58s **and 1.4** @71s show the user flying the UAV towards respectively the walls, ceiling and ground. The relevant collision avoidance task increases in weight and desired velocity away from the object resulting in the UAV gradually coming to a hover even though the user gives full control input towards the obstacle. **Phase 1.5**, @95s, shows the reactive behaviour of the UAV towards an unknown dynamic obstacle (mobile wall) while the user gives no lateral inputs.

B. Phase 2: static marker tracking

In the second phase of the experiment (between 105s and 165s) a person takes the marker, brings

[1]https://youtu.be/5eTx6oSC4f0

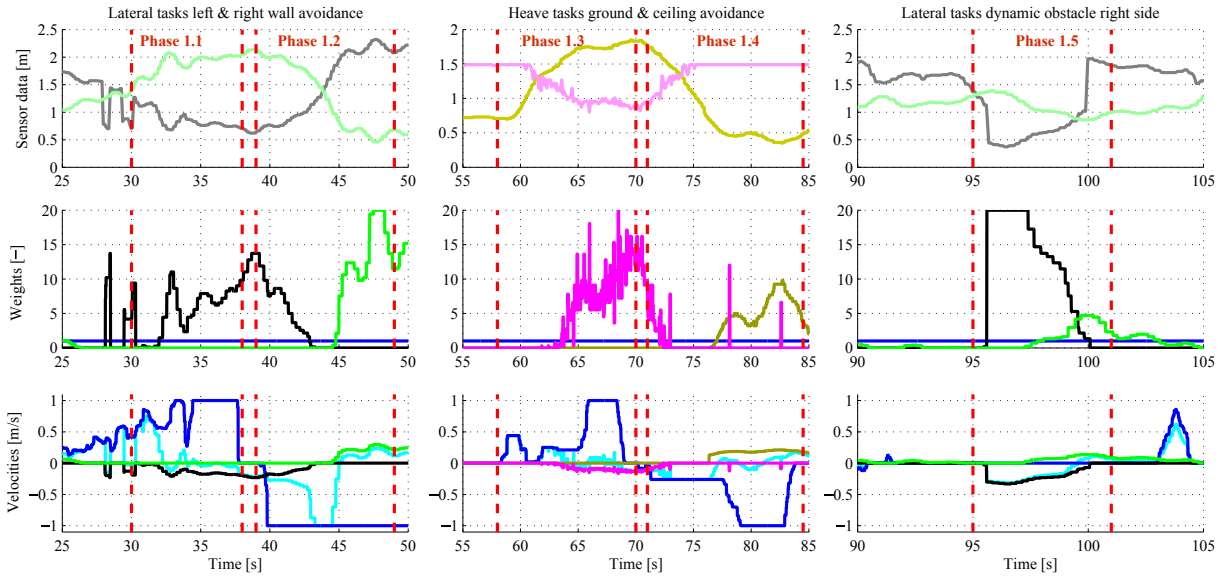


Fig. 5. Phase 1: Basic obstacle avoidance, no marker tracking. (legend: see Figure 7)

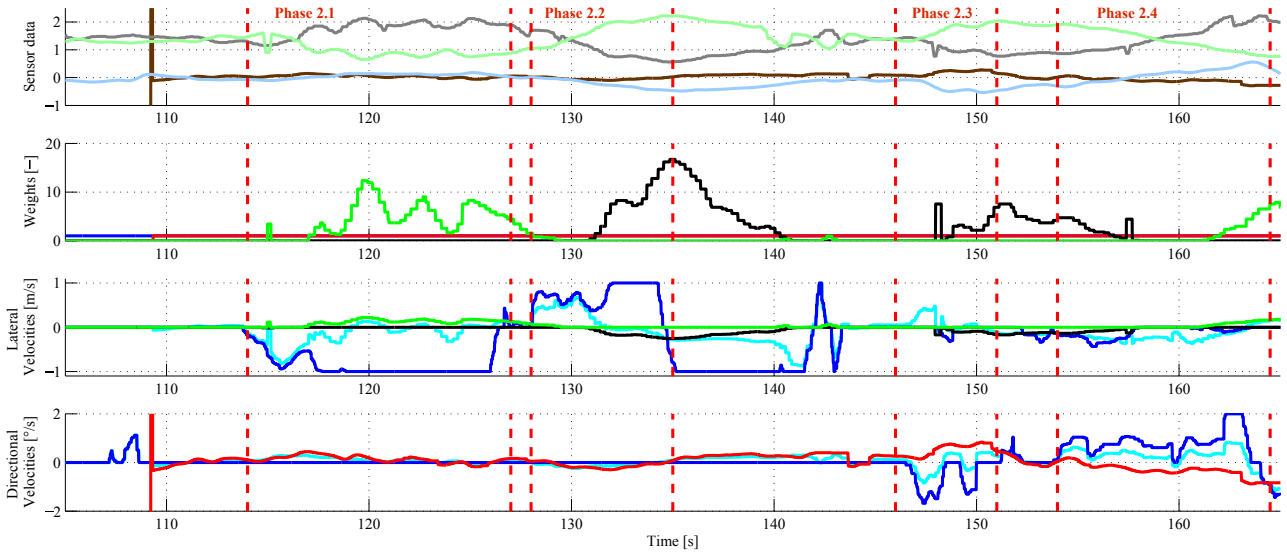


Fig. 6. Phase 2: Static marker tracking. (legend: see Figure 7)

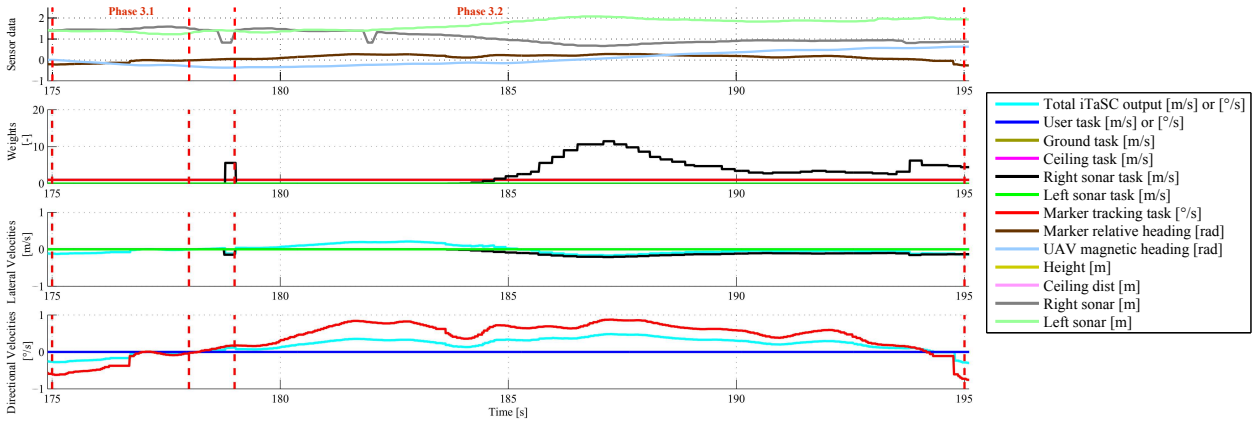


Fig. 7. Phase 3: Dynamic marker tracking.

it into view of the camera and keeps it stationary (Figure 6). The tracking task automatically activates (@109s) as soon as the marker is detected: the tracking task (red) weight becomes one. As the task's kinematic chain is based on a spherical coordinate system it measures the heading of the marker relative to the UAV's longitudinal axis (brown). The tracking task keeps the marker in the centre of the view through the UAV's lateral and directional control. Depending on the weighting with other tasks, it uses the lateral DOF more than the directional or vice versa. This phase of the experiment shows the composability of the tasks specified using different coordinate systems, equality and inequality constraints and the ability to (automatically) (de-)activate tasks at runtime.

Phases 2.1, @114s, **2.2a,** @128s, **2.3** @146 **and 2.4** @154 show the user commanding a lateral velocity in both directions forcing the tracking task to use directional control (yaw) to keep the marker centred and vice versa. **Phase 2.2b** @140s shows an unknown (and unexpected) dynamic obstacle introduced in the form of a **janitor** who suddenly walked through the test area. He was detected by the left sonar (grey), but the distance was large enough for its obstacle avoidance task (green) to only marginally affect the lateral velocity output (cyan).

C. Phase 3: dynamic marker tracking

The third phase (between 165s and 199s) shows the user giving no input, while the marker is moved left and right (Figure 7). This phase shows the UAV's smooth behaviour as the marker task is initially not constrained as the marker moves around, yet later on, when the right wall comes close and the tasks conflict in lateral direction, the tracking task uses yaw to track the marker similar to the user's input in phase 2.1.

Phases 3.1, @175s **and 3.2a** @179s show the marker (brown) moving respectively to the left and right resulting in the tracking task (red) to order a corresponding motion. This results in both lateral (roll) and directional (yaw) velocity outputs as no other obstacles are present, thus giving the tracking task free use of both DOFs.

During **phase 3.2b** @185s, the right wall comes so close that the right wall avoidance task (black) becomes active damping the tracking task in lateral direction thus bringing the UAV to a hover. As a result the tracking task uses yaw over roll to keep the marker in view.

VI. Discussion

The main advantages of our approach are combining intuitive shared control, obstacle avoidance and object tracking through real-time solving of composable tasks as a set of constraints.

Tasks can be physically interpreted, facilitating easy tuning of the control gains and choosing appropriate constraint weights for each task to achieve predictable overall behaviour of the UAV. An additional advantage is that the corridor's physical dimensions do not influence control gains and weights. Serving as an example of easy tuning, the marker tracking task was implemented after the other tasks were tuned and experimentally tested as a whole. Adding the marker tracking task only required to tune and test the task itself, which took under an hour. Moreover, the behaviour of the application as a whole remained predictable and interpretable.

The setup currently has two major limitations: the (lateral) velocity estimation and the near-hover flight speeds. The velocity is estimated by Kalman Filters (KFs) based on PixHawk's accelerometer and left and right sonar data. The velocity estimation requires a left or right wall in the vicinity at all times to remain accurate and the KF's accuracy is very sensitive to the accelerometer's calibration. In addition, the longitudinal control is currently fully under human control as obstacles are rarely present aft of the vehicle preventing good aft sonar measurements for correcting the accelerometer's drift resulting in an inadequate longitudinal velocity estimation. An optical flow camera could improve the velocity estimations given there's enough texture on the floor. The near-hover flight speeds limitation is due to iTaSC's desired lateral and longitudinal velocities being transformed to desired roll and pitch angles used by the PixHawk flight controller through a pure P-controller. This makes the UAV more susceptible to instability when rapidly changing pilot inputs are commanded. Currently, we are implementing a solution for this. However, this is of little consequence for the current use-case.

VII. Conclusions and future work

In this paper we proposed and experimentally validated a modular and generic constraint-based task specification approach and UAV flight control system architecture for human-machine shared control, capable of (dynamic) obstacle avoidance in unknown environments. We solve in (soft) real-time, i.e. at each time instance, a constrained optimization problem based on different tasks taking into account user intentions, dynamic obstacles and mission constraints, such as obstacle avoidance and visual marker tracking. It allows using various types of sensor data and space representations. Only onboard sensors and limited computational power were employed. The flight control system architecture provides fast, reactive behaviour and a more intuitive feedback, while retaining as much direct user control freedom as possible. However, this does not prohibit the approach to replacing (or assisting) the pilot with a global path planner. Moreover, our approach provides composable tasks that can be (de-)activated at runtime, while retaining physically interpretable results and easily tunable configuration parameters such as control gains and constraint weights. In future work we will add a global path planner and corridor detection image processing to assist/replace the human operator reducing his/her role to safety pilot creating full autonomous control with experimental validation, in outdoor conditions.

References

[1] J. Verbeke, D. Hulens, H. Ramon, T. Goedemé, and J. De Schutter, "The design and construction of a high endurance hexacopter suited for narrow corridors," in *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 543–551.

[2] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth, and B. Schiele, "Vision based victim detection from unmanned aerial vehicles," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 1740–1747.

[3] J. De Schutter, T. De Laet, J. Rutgeerts, W. Decré, R. Smits, E. Aertbeliën, K. Claes, and H. Bruyninckx, "Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty," *The International Journal of Robotics Research*, vol. 26, pp. 433–455, 2007.

[4] D. Vanthienen, T. De Laet, R. Smits, and H. Bruyninckx, "itasc software," 2015, http://www.orocos.org/itasc.

[5] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, pp. 1279–1291, 2012.

[6] W. Barfield and T. Dingus, Eds., *Human Factors in Intelligent Transportation Systems*. Psychology Press, 2014, ch. 3, pp. 55–94.

[7] F.A.A., "Unmanned aircraft systems (uas) frequently asked questions," 2015, https://www.faa.gov/uas/faq/.

[8] S. Anderson, "Constraint-based navigation for safe, shared control of ground vehicles," Ph.D. dissertation, Massachusetts Institute of Technology (MIT), 2013.

[9] A. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 3046–3052.

[10] D. Holz, M. Nieuwenhuisen, D. Droeschel, M. Schreiber, and S. Behnke, "Towards multimodal omnidirectional obstacle detection for autonomous unmanned aerial vehicles," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. XL-1/W2, pp. 201–206, 2013.

[11] J. Mendes, "Assisted teleoperation of quadcopters using obstacle avoidance," *Journal of Automation, Mobile Robotics & Intelligent Systems*, vol. 7, pp. 54–58, 2013.

[12] J. Roberts, T. Stirling, J. Zufferey, and D. Floreano, "Quadrotor using minimal sensing for autonomous indoor flight," in *European Micro Air Vehicle Conference and Flight Competition (EMAV)*, 2007.

[13] P. Stegagno, M. Basile, H. Bülthoff, and A. Franchi, "A semi-autonomous uav platform for indoor remote operation with visual and haptic feedback," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3862–3869.

[14] S. Hrabar, "Reactive obstacle avoidance for rotorcraft uavs," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 4967–4974.

[15] A. Kim and E. R., "Active visual slam for robotic area coverage: Theory and experiment," *The International Journal of Robotics Research*, vol. 34, pp. 457–475, 2015.

[16] I. Sa, S. Hrabar, and P. Corke, "Inspection of pole-like structures using a vision-controlled vtol uav and shared autonomy," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 4819–4826.

[17] C. Masone, P. Giordano, H. Bülthoff, and A. Franchi, "Semi-autonomous trajectory generation for mobile robots with integral haptic shared control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6468–6475.

[18] A. Mersha, S. Stramigioli, and R. Carloni, "On bilateral teleoperation of aerial robots," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 258–274, 2014.

[19] C. Ha and D. Lee, "Vision-based teleoperation of unmanned aerial and ground vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1465–1470.

[20] C. Samson, M. Le Borgne, and B. Espiau, Eds., *Robot Control, the Task Function Approach*. Oxford, Great Britain: Clarendon Press, 1991.

[21] W. Decré, H. Bruyninckx, and J. De Schutter, "Extending the itasc constraint-based robot task specification framework to time-independent trajectories and user-configurable task horizons," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 1933–1940.

[22] D. Vanthienen, "Composition pattern for constraint-based programming with application to force-sensorless robot tasks," Ph.D. dissertation, January 2015.